Journal of Location Based Services
Vol. 5, No. 2, June 2011, 79–99

Taylor & Francis
Taylor & Francis Group

Strategies for visualising 3D points-of-interest on mobile devices

Matthias Trapp*, Lars Schneider, Christine Lehmann, Norman Holz and Jürgen Döllner

Hasso-Plattner-Institute, University of Potsdam, Prof-Dr.-Helmert-Str. 2-3, Potsdam 14482, Germany

(*Received 1 December 2009; final version received 2 April 2011; accepted 6 April 2011*)

3D virtual environments are increasingly used as general-purpose medium for communicating spatial information. In particular, virtual 3D city models have numerous applications such as car navigation, city marketing, tourism and gaming. In these applications, points-of-interest (POI) play a major role since they typically represent features relevant for specific user tasks and facilitate effective user orientation and navigation through the 3D virtual environment. In this article, we present strategies that aim at effectively visualising POI in a 3D virtual environment used on mobile devices. Here, we additionally have to face the 'keyhole' situation, i.e. the users can realise only a small part of the environment due to the limited view space and resolution. For the effective visualisation of POI in 3D virtual environments we propose to combine specialised occlusion management for 3D scenes together with visual cues that handle out-of-frame POI. We also discuss general aspects and definitions of POI in the scope of 3D models and outline a prototype implementation of the mobile 3D viewer application based on the presented concepts. In addition, we give a first performance evaluation with respect to rendering speed and power consumptions.

Keywords: points-of-interest; 3D visualisation; mobile devices; real-time rendering

1. Introduction

Since mobile devices and gadgets have become an important part of our daily life, one can observe a growing number of mobile devices, such as Pocket PCs and mobile phones, as well as their application to car or pedestrian navigation systems. These devices have decreased in size, whereas their performance and feature set regarding location awareness and rendering capabilities have been improved. Based on these capabilities, mobile navigation and gaming starts to shift from the second into the third dimension. Navigation systems of companies such as Navigon or Sony start to provide users with 3D scenarios instead of flat maps. In this context, the effective visualisation of routes and landmarks (LMs) within 3D virtual environments (VE) on mobile devices represents a strong demand.

*Corresponding author. Email: matthias.trapp@hpi.uni-potsdam.de

ISSN 1748–9725 print/ISSN 1748–9733 online

DOI: 10.1080/17489725.2011.579579
http://www.informaworld.com

An important aspect of such systems is the visualisation of locations or points that have a particular priority during the navigation and orientation task. In the context of 3D geovirtual environments (3DGeoVE), such as 3D virtual city or landscape models, these are often denoted as *points-of-interest* (POI) or LM. The techniques to visualise such locations have to deal with two main cases: a point is visible within the current viewport (*on-screen*) or it is invisible, because it is located outside the viewport (*off-screen*). In case of 2D visualisation on-screen POI are often displayed as small icons overlaying a map. The problem of visualising 2D off-screen POI on mobile devices was successfully solved by introducing *partially-out-of-the frame* approaches (Baudisch and Rosenholtz 2003, Gustafson *et al*. 2008). Within 3D VE the effective and efficient visualisation of on-screen and off-screen POIs is still an open research question.

The visualisation of VE on mobile devices comprised a number of problems. Due to their nature, mobile devices suffer from small screen sizes and limited input capabilities. These are properties that are unlikely to change in future hardware generations. While the screen size tends to be constant, one can assume that the resolution of the display will shift from the QVGA to VGA standard or beyond in future version of mobile devices. Under these circumstances, the challenges are an appropriate visualisation of 3D scenes and objects, as well as to develop efficient navigation strategies and techniques that can be used to solve different routing tasks, e.g. finding the way to the nearest restaurant or similar within a city.

Within a 2D GeoVE, a user can efficiently compare distances to multiple locations, such as the distances to multiple restaurants, to estimate which of these is the nearest. For a complete estimation, all relevant objects have to be within the viewport. Due to zooming or panning, some of these objects may disappear into off-screen space. Especially for small-screen devices, the user is forced to frequently zoom in and out, and pan to see the off-screen objects. Consequently, within a certain zoom level, it is hardly possible to see all relevant objects, but only a subset of them, which makes spatial routing tasks more complicated and time-consuming. In addition to 2D visualisation, the visualisation of POIs within 3DGeoVE has to deal with the following challenges.

- **Occlusion:** Introducing the third dimension often leads to occlusion of important scene objects by occluding objects. This occurs especially in the pedestrian's view perspective, i.e. where the virtual camera is near or close to the ground. Especially in virtual 3D city models, only a small set of objects is visible due to a high degree of occlusion caused by building objects. For example, a restaurant cannot only be located off-screen, but can also be covered by other scene objects positioned in front of it, e.g. large buildings or skyscrapers. An effective visualisation for LMs should handle these object occlusions properly.
- **Projection and camera orientation:** Interactive visualisation of POI, especially those that are located off-screen, must perform effectively under varying camera orientations and types of projections. In particular, it has to handle the foreshortening occurring in classical perspective projections. The introduced size gradient of the objects, makes a distance estimations difficult for the user. Regarding the orientation of the virtual camera we can mainly distinguish between three settings: the pedestrian's, bird's eye

and finally, the plan view, which can be considered as analogue to a 2D setting.

- **Image-synthesis:** Due to the limited computing and memory capacity on mobile devices, a management of the scene geometry and textures is necessary. In terms of remote visualisation, current approaches can be classified into three main categories (Martin 2002): *client-side*, *server-side* and *hybrid* methods. The prototypical system presented in this article performs image synthesis on client-side.

Semantics of 3D virtual representations can be intuitively derived from real world objects by comparing them by means of reference points. These reference points reflect the main characteristics of the depicted objects and are needed to construct a relation between virtual object and real world object, according to the bilateral term of characters in the theory of semiotics (de Saussure 2001). If there are enough reference points, as usual for 3D representations, the interpretation can be handled in an efficient and effective way. Considering this as a crucial advantage of 3D space, the 2D Halo visualisation technique (Baudisch and Rosenholtz 2003) is supposed to be adapted within a 3D GeoVE.

For purposes of routing decisions and navigational tasks, this article introduces user awareness strategies (Figures 1 and 2), to emphasise 3D objects of virtual 3D city models that are located on and off-screen. These strategies based on 3D versions of the 2D Halo visualisation introduced by Baudisch and Rosenholtz (2003). The present work aims mainly at, but is not limited to, mobile devices as implementation platform. The four different prototypical 3D Halo visualisation are supposed to discuss and detect potentials and problems arising from transferring 2D partially-out-of-the-frame techniques to 3D. In addition thereto, we propose an extended POI notion that is especially suitable for 3D GeoVE. The introduced visualisation concepts for mobile navigation and gaming applications can be applied under changing perspectives. For maintaining the POI's visibility in 3D, we also focus on managing occlusions. Further, we briefly describe the implementation of our approach on a specific mobile device using a scene-graph based rendering system. Furthermore, we provide a performance evaluation with respect to the rendering

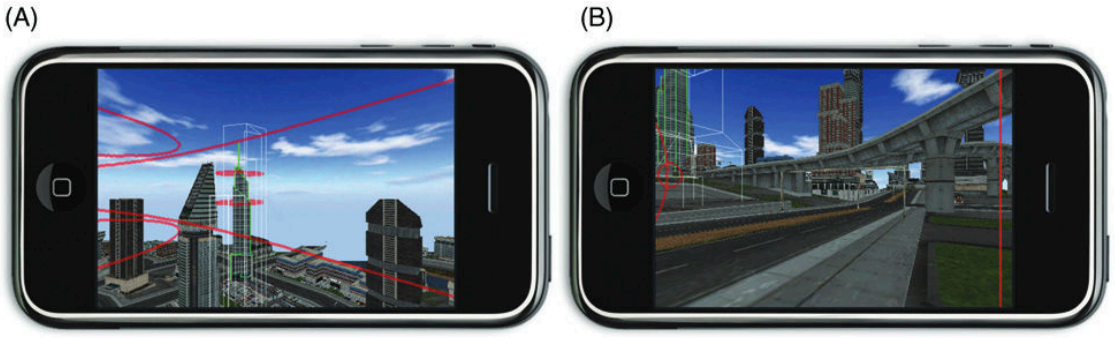

Figure 1. Examples of visualising POIs on a mobile device. (A) An object-space 3D Halo Circle visualisation combined with occlusion management showing three LMs. One LM is on the left front side out-of-view. One LM is farther away on the left side out-of-view and slightly behind the camera. One LM is right in front but occluded by another building (which is therefore rendered as wire-frame). (B) An image-based 3D Halo Projection visualisation. The LM is on the left side out-of-view (arc). One LM is marginally within the view (circle) but occluded by another building. One LM is in the right half-space behind the camera (straight line).

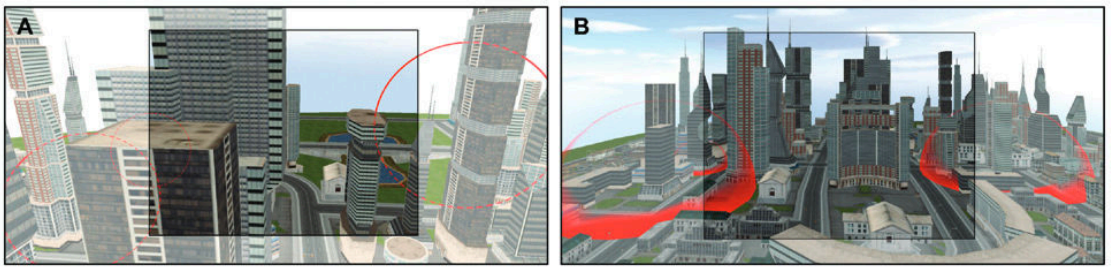

Figure 2. Mock-ups to clarifying the concept of 3D Halos located around three 3D scene objects classified as POIs. The saturated areas show the viewport visible to the user and the desaturated areas show the surrounding scene and POIs locations. (A) The 3D Billboard Halo approach (B) the 3D Sphere Halo approach for the same objects.

speed and power consumptions using virtual environments of different geometric complexity.

The applications of such visualisations are extensive. As GPS-enabled devices as well as software applications that use digital maps become more available, so too the applications for POI expand. Section 2 reviews related work on the topic of POI visualisation and interactive rendering on mobile devices. Section 3 introduces an extended POI notion and presents our approaches for their effective visualisation in 3D GeoVE. Section 4 gives a brief overview of the implementation of our rendering technique. Section 5 presents the results of our approach and discusses its performance as well as its problems and limitations. Section 6 concludes this article and present ideas for future work.

2. Related work

Geovisualisation deals with understanding our geography (Kraak 2002). Among others, this involves topics about the human perception and interpretation process concerning geospatial data. A number of tasks on geovisualisation are related to the specific application field of spatial data in public transport information systems (Dykes *et al.* 2005). This includes routing and navigation tasks with respect to POI, such as finding the nearest bus station or restaurant. Gustafson *et al.* (2008) as well as Burigat *et al.* (2006) classify off-screen awareness approaches into *Overview* + Detail and *Focus* + *Context* visualisation techniques.

Approaches for overview + detail visualisation an additional window or viewport that displays a survey knowledge of the current scenery. In the context of 3D GeoVE, simplified and minified views are common that show the scenery as plain 2D map. A major drawback of Overview + Detail is the additional cognitive task performed by the user to map between the information of both detail and overview screen (Chittaro and Burigat 2004, Gustafson *et al.* 2008). The second drawback, the map scale of the overview screen, is especially crucial when dealing with large-scale virtual environments (Chittaro and Burigat 2004): using a large map scale an overview map might not cover all POIs, whereas using a small map scale often omits too much detail, which could hamper the users' orientation. Further, an overview screen requires additional screen space that is limited on mobile devices (Gustafson *et al.* 2008).

Focus + context visualisation techniques do not rely on additional windows and visualise all information in a single view. All objects within the view centre are in focus and all surroundings represent the context. Gustafson *et al.* (2008) further distinguish between distorted views and non-distorted, contextual views. Employing a distorted view, objects in focus are visualised clear and sharp whereas the context is distorted. The transition between focus and context areas is typically smooth. A popular visualisation is the *fisheye view* (Sarkar and Brown 1994). Such visualisations have disadvantages in targeting or re-visitation tasks, due to the distorted visualisation (Hornbæk and Frøkjær 2001). Hence, these views are finitely suitable for spatial related task as they occur in 3D GeoVE. Non-distorted, contextual views do not aim to visualise all context information. Non-distorted, contextual views visualise POIs with abstract shapes that are overlaid on screen. For an off-screen POI an abstract shape acts as proxy to visualise the POI's existence. In virtual environments these proxies indicate traditionally direction and distance of the associated POI. Popular visualisation techniques are Arrows (Burigat *et al.* 2006), *City Light* (Zellweger *et al.* 2003), and *EdgeRadar* (Gustafson and Irani 2007).

The concept of partially-out-of-the-frame visualisation exploits the human visual system for proxy recognition and interpretation. All approaches described above, rely on an explanation or legend for the user to interpret the abstract shape. These explanations require either screen space or have to be memorised by the user. To overcome these problems, Baudisch and Rosenholtz (2003) introduced the 2D *Halo* visualisation. Distance and direction are implicitly conveyed by circles around POIs, that reach into the viewport. These partly visible circles are automatically completed through *amodal completion* conducted by the human visual system. The user gets an intuitive imagination of the POI's distance and direction. This *partially-out-of-the-frame* approach is borrowed from cinematography (Marsh and Wright 2000). The benefit is an intuitive and efficient visualisation of distance and direction.

Nevertheless, for a high number of POIs, the Halo visualisation suffers from cluttering induced due to overlapping arcs. To reduce cluttering Gustafson *et al.* (2008) introduced *Wedge* visualisation. A Wedge is an acute isosceles triangle in which the tip coincides with the off-screen POI and the two other corners are on-screen. Consequently, the triangle legs are partly visible and the user is able to determine distance and direction of the associated POI. The overlapping is reduced by the distance- and direction-independent adjustment of the Wedge's opening angle and its rotation around the POI. However, Wedges suffer from overlapping and cluttering, too, if the number of POIs is further increased.

Besides the off-screen awareness, visualisation of POIs has to deal with the problem of occluded on-screen POIs. In Elmqvist *et al.* (2008) are identified, several occlusion management patterns for reoccurring occlusion problems based on their taxonomy of occlusion management techniques.

3. Approaches for POI visualisation

This section presents four visualisation approaches (Figure 3) that can be applied to emphasise an off-screen POI within a 3D GeoVE. All techniques attempt to transfer the 2D Halo concept described in Section 2 and therefore also rely on on-screen proxy objects, which represent the off-screen POI within the view frustum.

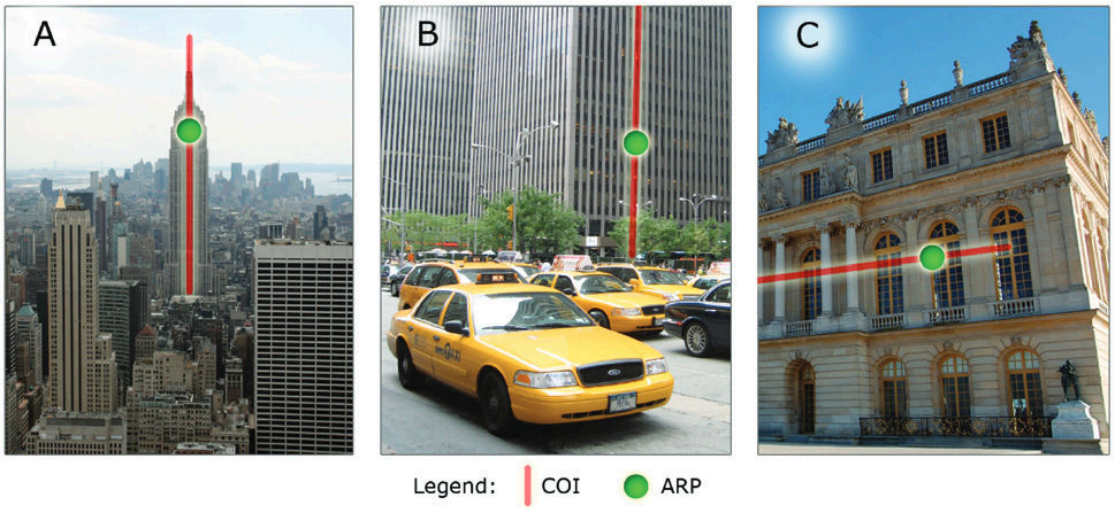

Figure 3. Comparison of four strategies for visualisation of a single POI that is located off-screen: 3D Sphere Halo (A), 3D Circle Halo (B), 3D Billboard Halo (C) and 3D Projection Halo.

While designing visual cues for 3D off-screen locations we proceed according to the following ambition: the main objective is the users intuitive understanding of distance and direction of an off-screen POI relative to the position of the virtual camera. Since every off-screen POI is visualised by means of an on-screen proxy object, it is important to enable the user to be aware of this relation. In this context it is especially difficult to design a proxy that communicates a POI behind the camera. In case of more than one POI, a proper differentiation between the respective on-screen proxies is required. An additional task is to ensure that these proxies are not significantly occluded by scene objects or other proxies. Furthermore, these proxies have to have as less viewport coverage as possible to minimise visual clutter and thus maximise the users' perception of the currently visible scene.

3.1. *Dynamic anchor points*

While experimenting and implementing different 3D on-screen and off-screen visualisation techniques, we discovered a lack of precision in the term POI, which is further elaborated in this section. At first we highlight different perceptions of the term found in literature. Afterwards, we propose a more precise definition suitable for visualisation of POIs in 3DGeoVE, especially virtual 3D city models.

With respect to 3DGeoVE, the notion POI or way point usually describes a 3*D point* or *object* that is part of this virtual environment, and which is of a particular interest for the user. In GPS navigation software systems, a POI is usually specified by latitude and longitude coordinates, at minimum. The classification of scene objects into either *interesting* or *not interesting* is performed with respect to a given context. In case of 3D GeoVE, these are 3D objects of a specific feature type or semantic, such as buildings, roads, or street furniture. Here, the context is usually navigation and a POI can be considered as navigation target or way point. Existing approaches for off-screen location visualisation, such as Arrows, Halos or Wedges

therefore require an exact point representation for every object to be applied correctly. Vendors of navigation systems, often store a fixed 2D georeferenced position (e.g. in WGS84) for each POI, such as the access and exit points for a gas station. Other objects, like as complex buildings, are usually represented by its geometrical centre point.

Mostly, this is sufficient in case the size of the presented object is relative small to its context, and has a convex shape and no major extend along one axis. In 2D space, a circle can reasonably approximate the shape of the object in most cases. Examples on a 2D map are the position of a person, a road intersection, or a house. But given an arbitrary position of a dynamic virtual camera, objects with specific extend along one axis, e.g. a skyscraper or a bridge, are not sufficiently represented by a static geometrical centre point. In 3D space this problem becomes more difficult due to the additional height dimension because every shape of an object must then be reasonably approximated, e.g. using a sphere. Objects with an appropriate point representation in 2D space, e.g. a skyscraper with quadratic basement for instance, have no such point in 3D space that can be used as reference for its off-screen visualisation.

To handle the above problem, we propose the terms *Appropriate-Representation-Point* (ARP) and *Cloud-of-Interest* (COI) to specify a POI with respect to 3D VE. An ARP is a geometrical point that represents an 3D object at its best, according to a given scene and configuration of the virtual camera. A COI is a set of all possible ARPs (P_i) of an object. We further introduce two functions: δ and σ. The function δ computes a COI for a given object. The function σ selects an ARP out of an object's COI for the scene and virtual camera:

$$\delta(Object) = \{P_1, P_2, \ldots, P_n\} = COI_{\text{Object}}, \quad P_i \in \mathbb{R}^3 \tag{1}$$

$$\sigma(scene, activeCamera, COI_{\text{Object}}) = ARP \in COI_{\text{Object}} \tag{2}$$

The two functions δ and σ can be application specific, changed at runtime, and can be adapted accordingly. For example, in the context of a 3D virtual city model, a COI for an object can be defined as a line along one of its main axis: for an object with a principal horizontal shape, a horizontal COI through its centre can be selected (Figure 4). For more complex object shapes, such as buildings with non-convex footprints, the computation of the respective COIs can be performed in a preprocessing step using straight skeleton methods (Barequet *et al.* 2008) or curve skeleton approaches (Sharf *et al.* 2007) for 3D shapes.

Considering the reference coordinate-system of a proxy, we can classify our visualisation techniques into *object-space* and *screen-space approaches*. Section 3.2 presents the object-space solutions *3D Halo Sphere*, *3D Halo Billboard* and *3D Halo Circle*. Section 3.3 covers the *3D Halo Projection* visualisation as special case of a screen-space approach. We further combine these techniques with a simple form of occlusion management for POIs that are are on-screen, but occluded by less important objects (Section 3.4).

3.2. *Object-space approaches*

This section presents the design and concept of three object-space approaches: *3D Halo Sphere*, *3D Halo Circle* and *3D Halo Billboard*. The first is a straight-forward

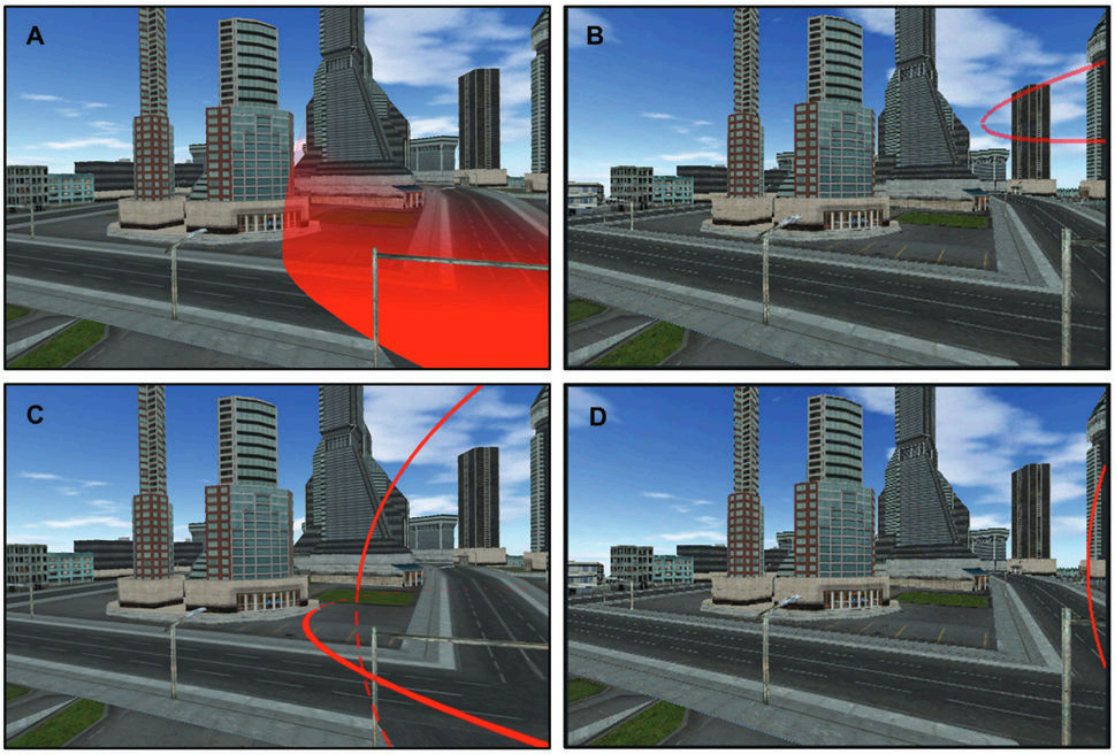

Figure 4. A (bird's eye view) and B (pedestrian's view) visualise two different ARPs, selected from the same COI, respectively the same object. The skyscraper's COI (A,B) is a straight line from the centre of the basement to the centre of the rooftop (main axis of the building). In contrast to that, C depicts a COI and an ARP of a long and flat building.

approach extending the 2D Halo (Baudisch and Rosenholtz 2003) to the third dimension by using a sphere. The circle approach simplifies this using only a 2D circle in 3D object-space approximating the sphere with respect to a reference plane that approximates the digital terrain model locally. To add additional cues, this concept is extended by the billboard approach that adds an additional approximation in the vertical plane.

3.2.1. *3D Halo sphere*

This approach visualises off-screen locations by means of a sphere (Figure 3(A)). To depict a Halo Sphere, two parameters, centre and radius, are required. We use the midpoint $B \in \mathbb{R}^3$ of the object's axis-aligned bounding box (AABB). The AABB represents the COI in this case. The point B is projected on the reference plane, which leads to $C \in \mathbb{R}^3$. Using C as the centre of the Halo sphere, C represents the ARB. We now compute the radius $r \in \mathbb{R}$ by means of the distance $d \in \mathbb{R}$ from C to the nearest plane of the view frustum.

In the following, we consider four planes of the view frustum to be relevant for computation: the left, right, bottom and top plane. Objects that are behind the camera (near plane) result in a Halo wherein the camera is located. For this reason, the screen would be completely covered by the Halo's back face. As a result, near plane and far plane are not considered in this computation. A distance d is computed by the Euclidean distance to the plane (of the four relevant planes E_{left}, E_{right}, E_{bottom}

and E_{top}), where the off-screen object is behind. Therefore, we compute the signed distances sd_p to all four planes and obtain d as follows:

$$d = \sqrt{\sum_{sd_p < 0} sd_p{}^2} \quad p \in \{left, right, bottom, top\} \tag{3}$$

These planes are defined according to their base point P and the normal vector $\vec{n}$. Only setting the radius as d would make the Halo to be tangent to the visible area. Hence, d has to be increased by a value that we denote as $\epsilon \in \mathbb{R}^+$: $d = d \cdot \epsilon$. We obtain the following parameterisation for a Halo H:

$$H = (\mathbb{R}^3, \mathbb{R}) = (C, r) = (C, d + \epsilon) \quad r, d, \epsilon > 0 \tag{4}$$

3.2.2. *3D Halo circle*

This technique visualises off-screen locations by means of drawing a circle around an ARP that is parallel to a reference plane, which approximates the virtual 3D city or landscape model. The circle is rendered around a vertical line through the ARP (Figure 3(B)). The ARP is selected with a function σ that selects a point out of the COI based on the minimal distance to the view frustum centre. The radius for both circles is determined as follows: At first, a line $\overline{P_A P_C}$ between ARP P_A and view frustum centre P_C is probed against all frustum planes to find an intersection (Figure 5(A)). In case the ARP is within the frustum, hence (potentially) visible, no intersection has occurred. Otherwise, an intersection point P_I is found. Using this point and frustum intrusion depth $\epsilon \in \mathbb{R}^+$, the circle radius r is determined by the formula: $r = Euclidiandistance(\overline{P_A P_C}) + \epsilon$.

3.3.3. *3D Halo billboard*

This approach extends the 3D Circle Halo by using of crossed billboards (Figure 3(C)). Billboards are used in various applications to depict complex geometry in real-time systems. A classic example for using billboards is the visualisation of vegetation objects in 3D landscape models models. Using billboarding, a complex 3D object, e.g. a tree, is replaced by an impostor texture. The orientation of a billboard is adjusted in the way that it faces some target, usually the camera position (Akenine-Möller *et al.* 2008). We use billboards to improve the visibility of the 3D Circle Halo and reduce visual cluttering of 3D Sphere Halo. To depict a Halo as a textured billboard, three parameters are required: centre, radius and camera position. Center and radius are computed analogue to the Halo Spheres. The camera position is necessary to align the billboard according to the user's position. To minimise occlusions between Halo Billboards, two different line styles are applied to the billboard. Regarding the portion of a Halo that is not occluded by other objects, the outline of the circle is continuously depicted. In case of occluded portions of the Halo, the circle's outline is depicted with a dotted, more transparent ($\alpha = 0.4$) line. Except for full occlusion (In this case only the dotted line is depicted and depth cues are missed), a Halo Billboard provides depth cues while it is visible in spite of occlusions.

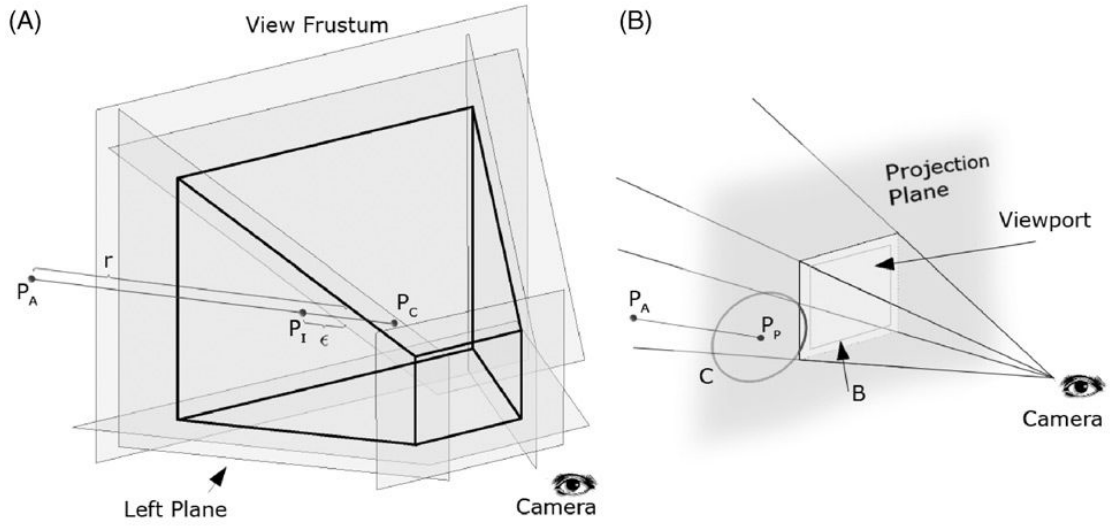

Figure 5. (A) The radius determination of a 3D Halo Circle (the resulting circles are omitted due to clarity). (B) 3D Halo Projection technique is illustrated. The circle C around P_P reaches into the area B, limited by an intrusion border, to help the user estimate the out-of-view distance and direction of P_A.

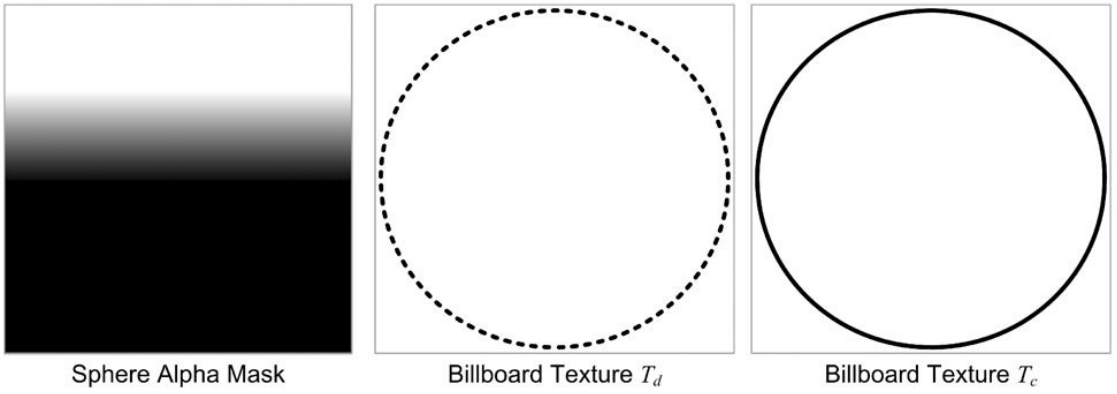

Figure 6. Textures used for stylisation of 3D Sphere Halo and 3D Billboard Halo. The alpha mask is used in combination with the vertex color to obtain an alpha-gradient for Spherical Halos. The texture T_d containing a dotted line style is used to communicate occluded parts of a 3D Billboard Halo while a continuous line style T_c is used otherwise.

The different line styles are implemented by using two textures (Figure 6). One texture shows the continuous line style, the second one shows the dotted, transparent line style. Thereby, the sampled colour C_{T_c} of T_c replaces the colour C_{T_d} of T_d in case there is an occlusion with respect to this fragment. The Halo Billboard can be augmented by adding the footprint. The footprint is depicted as a horizontal circle that is also rendered with the dotted and continuous texture.

3.3. *Screen-space approach*

The 3D Halo Projection approach visualises off-screen locations with a circle around the projection of an ARP (Figure 3(D)) on the planar projection surface. This

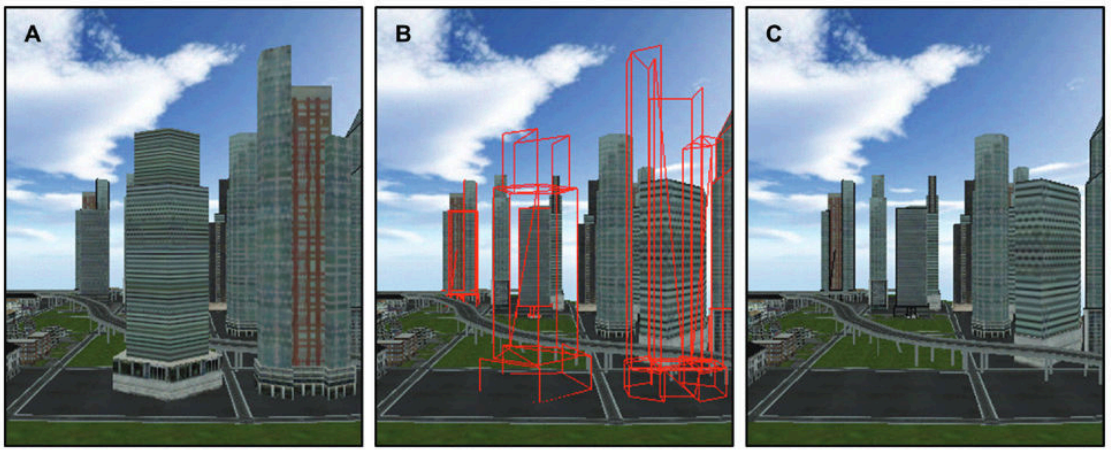

Figure 7. Occlusion management applied to building objects. The foreground building occludes a LM in the background (A). The occluder building is rendered using a wire-frame style (B) or is completely removed from the scene (C) to keep the LM visible.

reduces the 3D space problem to a 2D space problem. The point with the minimal distance to the Point-of-View is selected as ARP and is projected onto the projection plane. In case the projection point P_P is outside of a defined viewport border B, a circle C is drawn around P_P with a radius large enough to tangent B. Based on the visible arc, a user is able to estimate, i.e. the out-of-view distance and direction of the ARP (Figure 5(B)). This notion of distance is a major difference to 2D Halo visualisation and 3D object-space approaches. The 3D Halo Projection distance notion gives a user an impression how far a certain object is out-of-view.

3.4. *Occlusion handling*

Occlusion is a major problem when adapting the Halo concept to 3D-space, especially if a 3D scene is depicted from a pedestrian perspective. In the context of this work, there are basically four types of occlusions that are relevant for the presented object-space approaches.

3.4.1. *Object-object occlusions*

To compensate this type, we apply an occlusion management approach to counterbalance this phenomenon. Therefore, our approach turns occluding surfaces invisible or semi-transparent in order to maintain POIs visibly. This concept is summarised by the *Virtual X-Ray Pattern* (Elmqvist *et al.* 2008). This pattern is usually implemented using image-based approaches, whereas occlusion is determined on a per-pixel basis. We manage occlusion in object-space by omitting the rendering of the occluder or changing its rendering style, e.g. to wire-framed (Figure 7).

3.4.2. *Halo-object occlusions*

A Halo can occlude other objects, e.g. buildings. Our approach solves this problem by means of transparency. Thereby, the Halo is rendered with a transparency

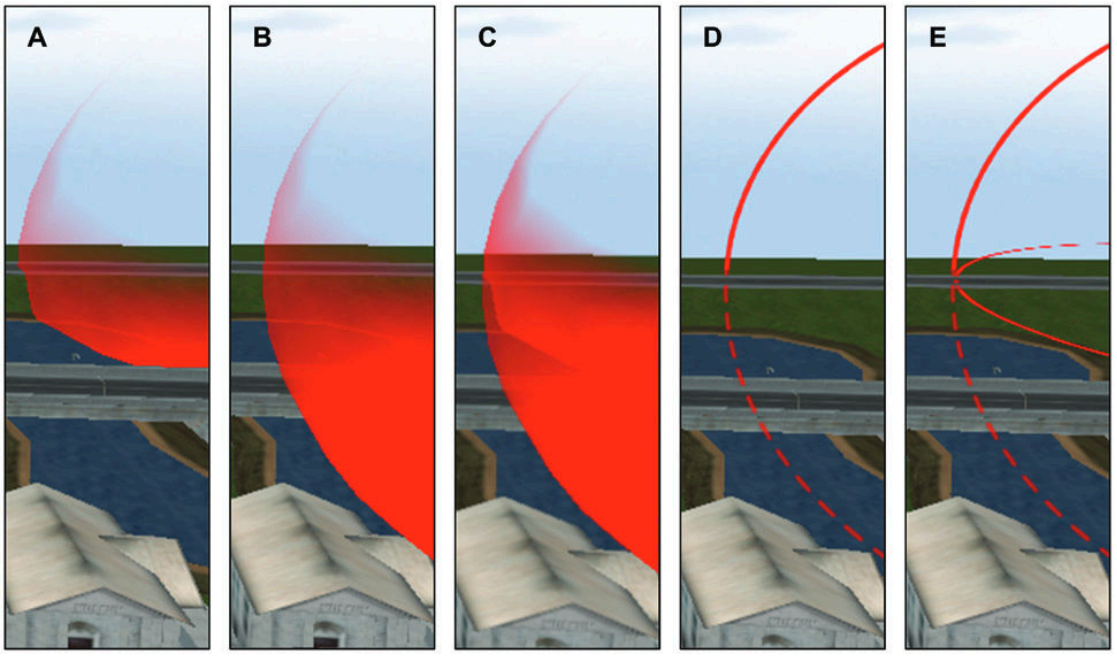

Figure 8. Comparison of five variants for 3D Halos. Spherical Halos with normal depth mode (A), foreground depth mode (B) and mixed depth mode (C). Billboard Halos without footprint (D) and with footprint (E).

gradient varying between 1 for the Halo's outlines and 0 for the inside area. In case of a Halo Sphere, a semi-transparent appearance is chosen to reduce occlusions regarding objects behind the Halo. To obtain this transparency, the sphere is textured using an alpha-mask texture and a colour texture.

3.4.3. *Object-halo occlusions*

Halos representing an off-screen POI that is positioned in the background can be occluded by front objects. With an increasing distance between off-screen POI and camera, the portion of a Halo that is occluded may increase until it becomes invisible. Hence, the Halo becomes the occluder, which corresponds to the worst case. The disadvantage is the loss of depth cues.

3.4.4. *Halo-halo occlusions*

As the user takes a larger set of relevant POIs into account, off-screen objects may become closer. As a result, intersections and overlaps of Halos increase. This can lead to visual clutter that makes the perception and interpretation of the Halos cues more difficult. Intersections between Halo Spheres occur between lateral areas of the sphere. In conclusion, they lead to occlusions of three dimensional complexity. To handle these occlusions, the Halo Spheres variant implements three different depth modes (Figure 8(A)–(C) and Figure 9(A)–(B)):

- **Normal depth mode:** For this depth mode, the Halo's sphere is rendered with normal depth test whereas fragments having a depth lower than the depth buffer's value are rendered. Hence, the sphere appears partially visible since the lower half of the sphere is under the referencing plane. The upper half

can be additionally occluded by objects in the foreground, e.g. buildings in a city model.

- **Foreground depth mode:** The sphere is namely rendered with writing to depth buffer but turning off the depth test. As a result, the sphere appears in foreground and looses depth cues.
- **Mixed depth mode:** This mode implements a combination of the previously presented modes. It depicts the spheres that are rendered with normal depth mode as well as the spheres in foreground. Hence, the spheres in foreground overlay the normally rendered sphere and a terrain intersection area becomes visible to give depth cues.

4. Implementation for real-time rendering

4.1. *Interactive rendering on mobile devices*

Playing 3D games, exploring, and navigating through 3D virtual environments is already possible on mobile devices (Chehimi *et al*. 2005). Due to the limitations in screen size, battery power and rendering hardware, the obtainable rendering performance is not comparable to those of modern desktop systems. Consequently, visualisation applications developed for desktop environments do not scale well for mobile devices (Chittaro 2006).

However, the m-LOMA project Nurminen 2006 presents a 3D engine that performs rendering of 3D city models at interactive rates from any viewpoint, discuss optimisation principles, a system architecture, as well as describes a user study. 3D mobile maps provide realistic visualisation of objects, a high degree of free movement and a volumetric representation of space. The assets and drawbacks of 3D maps in comparison to 2D maps, which are commonly installed on modern mobile devices, are presented by means of a user study and discussed in Oulasvirta *et al*. (2008). In Moser and Weiskopf (2008), a rendering technique for interactive direct volume visualisation on mobile devices is presented and limitations of mobile graphics devices are discussed.

Third party development on mobile devices is enabled by device specific Software Development Kits (SDKs) that provide typically a high abstraction to the device hardware and the windowing system. Popular examples are Java ME Wells (2004), Android SDK DiMarzio (2008), and the Apple iPhone SDK Mark and LaMarche (2008).

Our exemplary implementation is based on the Apple iPhone SDK and the iPhone as testing device, but can be implemented on other mobile devices having the same rendering capabilities. As a state-of-the-art multimedia device, it is capable of rendering 3D graphics fully hardware accelerated using a dedicated graphics processing unit (GPU) and possesses a capacitive multi touch screen. The hardware accelerated rendering is accessed through the low-level Application Programming Interface (API) *OpenGL ES*, which represents a subset of OpenGL (Board 1999) especially suited for mobile devices.

4.2. *Basic rendering technique*

Based on OpenGL ES, a rudimentary scene-graph system has been implemented to provide the scene management facilities for the proposed visualisation as depicted

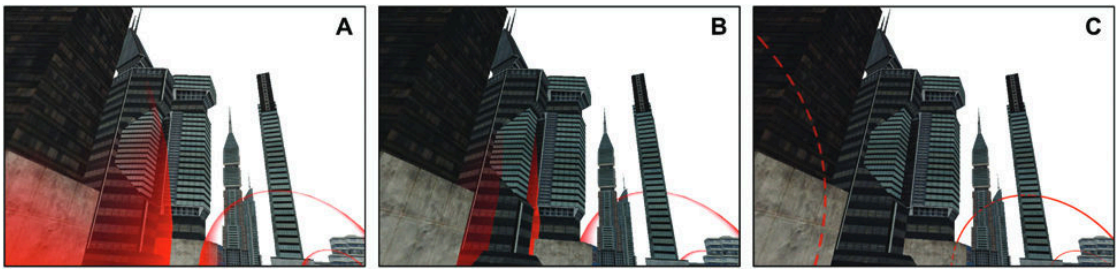

Figure 9. Screenshots for 3D Spherical and 3D Billboard Halo from a pedestrian's view perspective. It shows two off-screen POIs beneath the camera and one on the left side. (A) 3D Sphere Halo using the foreground depth mode; (B) the normal depth mode and (C) the 3D Billboard variant for the same setting.

in Figure 10. The rendering system performs the image-synthesis in-core, i.e. no out-of-core streaming approaches are used. This implicitly makes the assumption that the complete virtual 3D city model fits into device memory. However, the proposed prototypical implementation needs to be adapted to work for arbitrarily large models. In particular this concerns the 3D geometry and textures as well as the POI data.

The POI visualisation and occlusion management are encapsulated in so-called techniques that traverse the scene graph and process the scene objects accordingly. The traversal is implemented according to the Visitor pattern (Gamma *et al.* 1994). The pseudo-code in Algorithm 1 exemplifies the handling of the currently visited *object* within the scene graph *scene* and a given camera *activeCamera*.

Algorithm 1: Visualisation of POI

```
if viewFrustumCulling(object, activeCamera) then
  for all visible POI do
    if occlude(object, POI, scene, activeCamera) then
      cull(object) or changeStyleAndRender(object) {Section 3.4}
    else
      render(object)
    end if
  end for
 else if isPOI(object) then
  σ_object = selectSigmaFunction(object)
  δ_object = selectDeltaFunction(object)
  ARP = σ_object(scene, activeCamera, δ_object(object)) {Section 3.1}
  renderHalo(object, ARP)
end if
```

The technique pre-traverses the scene graph and sets the corresponding flags for the removal or wire-frame rendering of an object according to a preprocessed visibility test. These flags are considered in the subsequent rendering traversal and the scene objects are rendered accordingly. The visibility test determines for every visible object whether it occludes a POI or not. It is based on a ray-intersection test (Glassner 1990) between the AABB that is derived from the tested mesh and the rays constructed from the current view-point to visible POIs. We exclude the ground meshes from this visibility test.

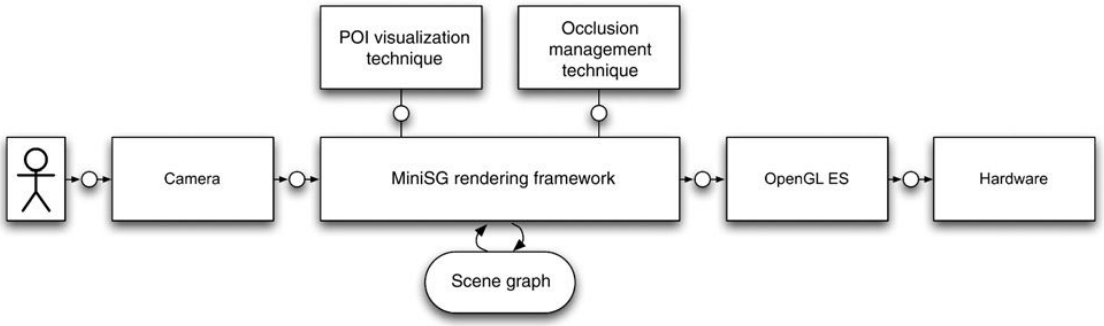

Figure 10. Schematic view of our rudimentary scene-graph system. A *User* controls a *Camera* and the rendering framework uses the *POI Visualisation Technique* and the *Occlusion Management Technique* to render the Scene Graph via *OpenGL ES* calls accordingly.

The visualisation techniques have to adapt the respecting parameters due to camera movement. Hence, the computation and rendering of Halos has to be performed per frame, based on the current view frustum. The rendering requires two rendering passes. Despite rendering the complete scene, the first pass is used to calculate centre and radii for each halo. In the second pass, the Halos based on the calculated parameters are added to the scene graph and rendered, calling method *renderHalo*.

4.3. *Performance evaluation*

We have tested our approach using two virtual 3D city models of different geometrical complexity. All meshes are textured by one of 18 textures, e.g. grass, buildings. Moreover, one ambient light, depth testing, back-face culling, view-frustum culling, and smooth shading are enabled (Figures 1, 11). All results are summarised in Table 1. Our implementation is able to render the described scenes at interactive frame-rates. The energy consumption, which emerges from rendering the described scenes, is measured indirectly through battery charge state. Consequently, the respective results can be considered imprecise.

Test 1 (T1) processes no OpenGL rendering at all, only the display is activated to measure a reference value for idle energy consumption. T2 renders a medium and large size model at the maximum frame-rate possible. As a result, the energy consumption roughly doubles, consequently the device runtime halves. T3 renders the same models with a limited frame-rate. As a result, the energy consumption decreases. This indicates a direct correlation between model complexity, rendering speed and the energy consumption.

4.4. *Preliminary user evaluation*

In order to present a preliminary verification we test Spherical Halos and Billboard Halos (without footprint) variants with 30 non-computer-specialists (22 male and 8 female). The age span varies between 25 and 50 years. Most of them have only minimal experience with 3D virtual environments. The test candidate sits in front of a computer screen. The testing environment is given within a 3D GeoVE containing

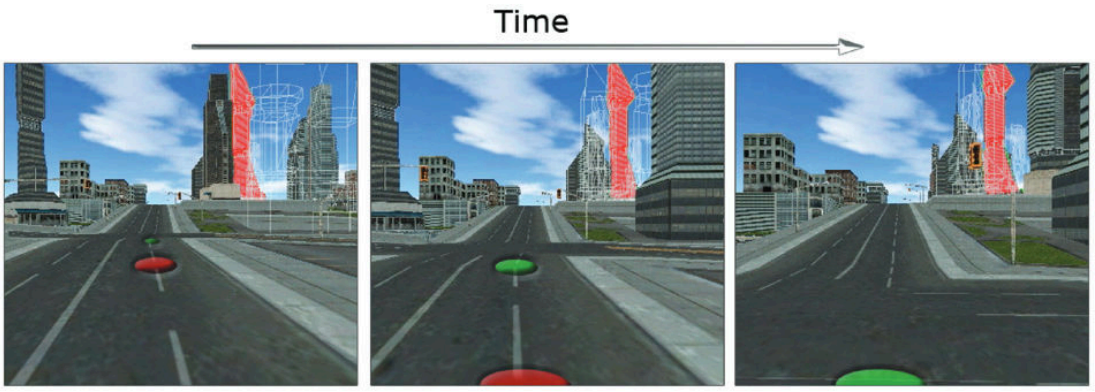

Figure 11. Occlusion management visualised over time. As the camera moves forward, all objects that occlude a POI (red) are rendered as wire-frame.

Table 1. Results of our performance evaluation. T1 is a reference test for the relative energy consumption. T2 renders the scenes at maximum frame-rate. T3 renders with a limited frame-rate that reduces the energy consumption.

Test ID	Model size	Texture data in MB	Meshes (avg. visible)	Triangles (avg. visible)	Avg. FPS (* limited)	Rel. energy consumption
T1		–	–	–	–	1.0x
T2	Medium	4.1	624 (24%)	12380 (35%)	23	2.3x
	Large	8.2	5924 (14%)	34596 (17%)	14	2.2x
T3	Medium	4.1	624 (24%)	12380 (35%)	10*	1.5x
	Large	8.2	5924 (14%)	34596 (17%)	10*	1.8x

a city model with buildings, streets, bridges and green areas. The test person selects three buildings using the mouse. Afterwards, the test advisor moves the virtual camera in a way that these buildings disappear into off-screen space and approximated using 3D Halos. The test candidate must identify the nearest of the three buildings with respect to the camera position.

The test criteria are efficiency and accuracy of the position estimation. In this context, accuracy means first to identify the correct building, which is the nearest, and second to name the correct order of the three buildings starting with the nearest. Thereby, the nearest building has a significant distance to the two other buildings. The distance between the second building and third building is small with respect to the first and second building. Furthermore, people can answer with naming a building and a building order respectively, or can state to do not know the answer, which also is counted as a failure.

The results of the preliminary test are depicted in Table 2. The values are rounded up to integer values. Regarding Spherical Halos, the test results show that the testing persons tried to integrate the Spherical Halos into the building scenery. For this reason, Spherical Halos do not achieve good results concerning perceptibility, i.e. they could be hardly interpreted as placeholders for off-screen objects.

Table 2. Preliminary user-test with 30 test persons: with Billboard Halos (without footprint) (B), more people can identify the nearest building than with Spherical Halos (S). As a countermove, the order of the other buildings is confused more family frequently with Billboard Halos than with Spherical Halos.

Task	Success for S	Success for B	Abstention
Identify nearest building (%)	75	83	None
Identify correct order (%)	68	37	7 (for B)

Concerning the task of identifying the correct order, two of 30 test persons did not know an answer using Billboard Halos (≈7%). The main limitation of Billboard Halos is the lack of accuracy since only one intersection point gives a depth cue.

5. Results and discussion

This section discusses the visualisation techniques with respect to the 'Halo's usefulness in tasks involving spatial reasoning' (Baudisch and Rosenholtz 2003). The question is whether this efficiency is lost due to adapting this technique for 3D GeoVEs.

The 3D Halo Circle approach enables the user to determine the off-screen POI distance and direction by extrapolation of a complete circle out of the partly visible circle curve (Figure 1(A)). POIs, which are on-screen, are also emphasised using the 3D Halo Circle approach. An asset of 3D Halo Circle is the visualisation of POIs behind the virtual camera. In such case, the user has the impression of being within the circle, which is consistent with the visualisation a user would naturally expect. The visualisation of more than one POI leads to visual clutter, due to the an increasing number of lines. The different circles will then cause a certain level of distraction and thus, it becomes difficult to distinguish between the circle pairs of a POI.

Moreover, as the circle radius increases, i.e. the POI is farther away, it becomes harder for a user to estimate the radius. Further, the circles occlude the scene objects (e.g. buildings), which is necessary to support the user's mental distance estimation within a distorted view frustum. Following to that, a user will probably not be able to mentally complete the radius in all cases, especially if too much of the circles area is occluded. Another disadvantage is the unbound viewport coverage. The circle can potentially be all over the viewport and, therefore, distract the user from perceiving the on-screen objects.

The 3D Halo Projection method enables a user to be aware of the distance and direction of the projection of a POI (Figure 1(B)). Since the direction is not altered due to projection the user is directly aware of the current POI direction. However, it is not possible to map the circle radius to the actual POI distance. Instead, the radius communicates only how far a POI is out-of-view. In case a POI enters the viewport, a small circle is drawn to indicate the relation between POI and circle. POIs behind the camera are visualised with a straight line on the viewport side of the respective half-space. A drawback of this approach is the possible distraction of the user induced

through the flipping of that line, caused by POI changing the half-spaces. The possible overlapping of lines, due to a high number of POI represents a further drawback of this approach. However, a major advantage of 3D Halo Projection is the limited viewport distraction, ensured by an intrusion area.

The 3D Halo Sphere enables the user to determine the off-screen POI distance and direction by estimating a complete sphere out of the partially visible sphere fragment. Using Halo Spheres, the problem of occlusions is expanded from lines to areas that intersect. In addition, Halo Spheres are self-occluding. Hence, the Halo Sphere is textured with an alpha-gradient-image having $\alpha = 1$ for the portion near the ground and $\alpha = 0$ for the upper portion of the Halo. Hence, the outline of the Halo is visible, while background objects stay perceptible. If the Halo is in foreground, there is no information anymore about depth, i.e. z-coordinates of the off-screen POI's position. As a result, the accuracy of the position estimation decreases. For this reason, the third mode of Halo Spheres combines the normally occluded Halo and the Halo in foreground.

3D Halo Spheres have the advantage to provide intuitive depth cues since a 3D terrain intersection curve is provided. However, in most cases they are not interpreted as placeholders for off-screen objects, but tried to be integrated as objects of the scene resulting in disorientation. Using the parameter ϵ, the size of the Halo can be adjusted. Thus, ϵ can be manually set or automatically derived depending on the distance to the camera. The limitation of using ϵ to resize Halos is a loss of accuracy. As the size is handled specifically for every Halo $H_i = (C_i, r_i)$, the Halo parameters have to be extended to $H_i = (C_i, r_i, \epsilon_i)$. As a result, the scales of the Halos are distorted, additionally to the distortions due to different scales of objects resulting from the perspective view.

Regarding the 2D Halo visualisation technique, an intrusion border is defined to guarantee that Halos do not take too much screen space. Thus, the Halo outline is at most tangent to this border. For 3D space, the intrusion border becomes an intrusion frustum, i.e. the border has to be extended for all planes of the view frustum since 3D Halos can reach from every direction into the visible area. The intrusion frustum can be figured as an inner view frustum. The computation of the Halo parameters now uses the intrusion frustum instead of the view frustum. As a result, the parameter ϵ would not be necessary anymore, since the intrusion frustum is already arranged inside the visible space. Hence, distortions are avoided due to uniform scaling of the Halos.

- ***3D Halo Billboards*** enables the user to determine the off-screen POI distance and direction by estimating a complete circle out of the partially visible circle fragment. The Halo Billboard combines a continuous, opaque circle for non-occluded regions with the dotted-lined, transparent circle in case of occluded regions.

Halo Billboards provide a good perceptibility since they are interpreted as separate meta-objects to the scene. As the Halo Billboard only shows the outline of circle, it minimises the occlusion problem. Due to intersections between lines, Halo Billboards are less complex and do not exhibit self-occlusions. The disadvantage of Halo Billboards is the imprecise depth cue, as the intersection of the circle's outline with the ground is only an intersection point. Together with the second intersection point of the circle that is constructed cognitively, there are only two points that are supposed to complete the terrain intersection area mentally. The impreciseness

results from the loss of information as only an approximated line can be constructed from one given intersection point and one cognitively constructed point (Figure 8(E)). In case a foreground object occludes the intersection point, this depth cue is also lost (Figure 8(D)). Regarding Halo Billboards with footprint, the depth cue can maintained, as the footprint is at least shown as a dotted circular outline in case of occlusions.

- ***Occlusion management*** ensures the visibility of POIs. To simply hide an occluder is a straight-forward approach to reveal the occluded POI. However, the information of the occluder's position, shape, appearance and its spatial context is lost. When the observer navigates through the 3D virtual environment with a certain number of POIs and occlusion situations, this technique introduces popping-artifacts, i.e. the occluders appear and disappear frequently. This effect might distract the user and decreases the overall visual quality.

  Rendering the occluder using a wire-frame style or similar can maintain the contextual information while minimising the popping effects to a certain degree (Figure 11). However, it increases the visual complexity and the cognitive load for the user, especially if several wire-frame objects are visible and overlap each other. This effect depends mainly on the geometric complexity of the 3D models, i.e. the number of triangles of the corresponding mesh. The more triangles the object consists of, the more lines are generated. Thus, in a situation with a high number of consecutively aligned occluders rendered in wire-frame style, the high number of visible lines can lead to occlusion in turn. The increased cognitive load, recognisable in a scene with a large amount of wire-frame objects, occurs in techniques employing semi-transparency as well (Elmqvist *et al*. 2008).

6. Conclusions and future work

This article presents four novel approaches for the visualisation of POIs in 3D geovirtual environments. Our concept is based on the combination of a partially out-of frame technique, which are adapted to 3D, and occlusion management. We further present an extended notion for characterising POIs in 3D virtual environments, which enables their selection on a per-object basis at runtime. The provided performance evaluation considers the rendering performance and the power consumptions of our implementation.

Our main task for future work is to compensate visual clutter introduced by the on-screen proxy objects. The circle-based visualisation can be improved by using stacks of circles to enable a visual separation for different POIs. In such case, different stacking schemes, which define what circle is above the other, as well as priority-based or distance-based schemes are possible. Based on the above extensions, we plan to conduct a comprehensive user evaluation. This would include the efficiency of our approach and a comparison of our 3D visualisation to alternative 2D off-screen location visualisation techniques, such as Overview+Detail techniques.

Acknowledgements

This work has been funded by the German Federal Ministry of Education and Research (BMBF) as part of the InnoProfile research group '3D Geoinformation' (www.3dgi.de).

References

Akenine-Möller, T., Haines, E., and Hoffman, N., 2008. *Real-time rendering*. 3rd ed. Natick, MA, USA: A. K. Peters, Ltd., 1045.

Barequet, G., *et al*., 2008. Straight skeletons of three-dimensional polyhedra. *In: ESA '08: Proceedings of the 16th annual european symposium on Algorithms*. Berlin, Heidelberg: Springer-Verlag, 148–160.

Baudisch, P. and Rosenholtz, R., 2003. Halo: a technique for visualizing off-screen locations. *In: CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, April 2003, New York, ACM Press.

Board, O.A.R., 1999. *OpenGL(R) reference manual: the official reference document to OpenGL, Version 1.2*. 3rd ed. Amsterdam: Addison-Wesley Professional.

Burigat, S., Chittaro, L., and Gabrielli, S., 2006. Visualizing locations of off-screen objects on mobile devices: A comparative evaluation of three approaches. *In: MobileHCI '06: Proceedings of the 8th conference on Human–Computer interaction with mobile devices and services*, September 2006, New York, ACM Press.

Chehimi, F., Coulton, P., and Edwards, R., 2005. Evolution of 3D games on mobile phones. *In: ICMB '05: Proceedings of the international conference on mobile business*. Washington, DC, USA: IEEE Computer Society, 173–179.

Chittaro, L., 2006. Visualizing information on mobile devices. *Computer*, 39 (3), 40–45.

Chittaro, L. and Burigat, S., 2004. 3D location-pointing as a navigation aid in virtual environments. *In: AVI '04: Proceedings of the working conference on Advanced visual interfaces*, May 2004, New York, ACM Press.

DiMarzio, J., 2008. *Android: a programmer's guide*. New York City: McGraw-Hill Osborne Media.

Dykes, J., MacEachren, A.M., and Kraak, M.-J., 2005. *Exploring geovisualization*. Amsterdam: Elsevier.

Elmqvist, N., Tsigas, P., and INRIA, O., 2008. A taxonomy of 3D occlusion management for visualization. *In: Visualisation and Computer Graphics*, January 2008, Piscataway, NJ, IEEE Educational Activities Department.

de Saussure, F., 2001. *Grundfragen der allgemeinen Sprachwissenschaft*, Berlin: Walter de Gruyter GmbH.

Gamma, E., *et al*., 1994. *Design patterns: elements of reusable object-oriented software (Addison-wesley professional computing series)*. Boston: Addison-Wesley Professional.

Glassner, A. S., 1990. *Graphics gems I (Graphics Gems – IBM)*. Waltham, Massachusetts: Morgan Kaufmann.

Gustafson, S., *et al*., 2008. Wedge: clutter-free visualization of off-screen locations. *In: CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, April 2008, New York, ACM Press.

Gustafson, S. and Irani, P., 2007. Comparing visualizations for tracking off-screen moving targets. *In: ChI '07: CHI '07 extended abstracts on Human factors in computing systems*, April 2007.

Hornbæk, K. and Frøkjær, E., 2001. Reading of electronic documents: the usability of linear, fisheye, and overview+detail interfaces. *In: CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM, 293–300.

Kraak, M.-J., 2002. Some aspects of geovisualization. *GeoInformatics*, 5, 26–37.

Mark, D. and LaMarche, J., 2008. *Beginning iphone development: exploring the iphone SDK*. New York City: Apress.

Marsh, T., Wright, P., 2000. Using cinematography conventions to inform guidelines for the design and evaluation of virtual off-screen space. *In: AAAI 2000 spring Symposium series Smart Graphics*, January 2000, Menlo Park, USA, AAAI Press, 123–127.

Martin, I.M., 2002. Hybrid transcoding for adaptive transmission of 3d content. *In: IEEE International Conference Multimedia and Expo (ICME)*, Vol. 1, August 2002. Hawthorne, NY, USA: (IBM, IEEE CS Press) Thomas J. Watson Research Center, 373–376.

Moser, M. and Weiskopf, D., 2008. Interactive volume rendering on mobile devices. *In: Workshop on vision, modelling, and visualisation VMV '08*. Berlin: Max-Planck-Gesellschaft, 217–226.

Nurminen, A., 2006. A platform for mobile 3D map navigation development. *In: Mobile HCI '06: Proceedings of the 8th conference on Human–Computer interaction with mobile devices and services*. New York, NY, USA: ACM, 101–104.

Oulasvirta, A., Estlander, S., and Nurminen, A., 2008. Embodied interaction with a 3D versus 2D mobile map. *Personal and Ubiquitous Computing*, 13 (4), 303–320.

Sarkar, M. and Brown, M., 1994. Graphical fisheye views. *Communications of the ACM*, 37 (12), 73–84.

Sharf, A., *et al*., 2007. On-the-fly curve-skeleton computation for 3d shapes. *In: Eurographics 2007 (computer graphics forum)*, Vol. 26. Vol. 26, Prague: Blackwell, 323–328.

Wells, M. J., 2004. *J2ME game programming (game development)*. Florence, USA: Course Technology PTR.

Zellweger, P., *et al*., 2003. City lights: contextual views in minimal space. *In: CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, April 2003, New York, ACM Press.